PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 1/00 — A1

(11) International Publication Number: WO 00/54128

(43) International Publication Date: 14 September 2000 (14.09.00)

(21) International Application Number: PCT/US00/06242

(22) International Filing Date: 10 March 2000 (10.03.00)

(30) Priority Data:
09/267,269 12 March 1999 (12.03.99) US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application
US 09/267,269 (CIP)
Filed on 12 March 1999 (12.03.99)

(71) Applicant *(for all designated States except US)*: CURL CORPORATION [US/US]; 8th floor, 400 Technology Square, Cambridge, MA 02139–3539 (US).

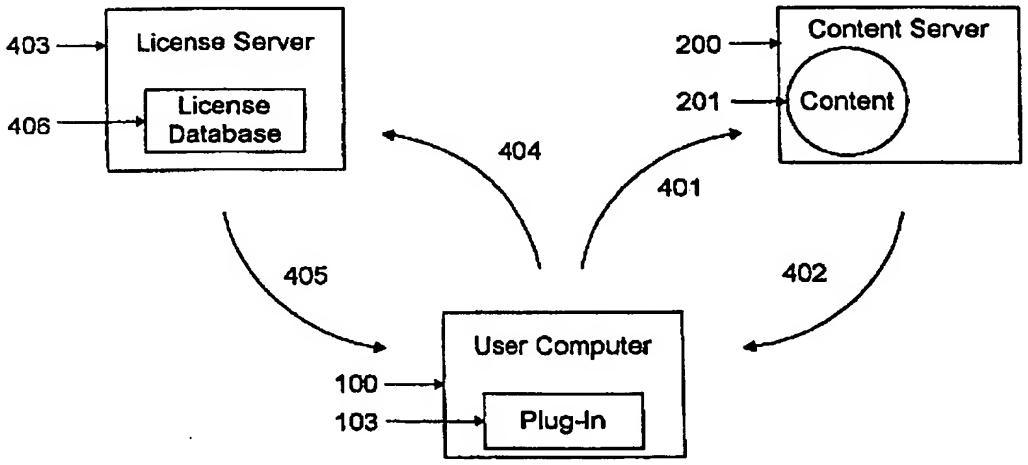
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: WARD, Stephen, A. [US/US]; 199 Coolidge Avenue, Watertown, MA 02172 (US). MAZER, Murray, S. [CA/US]; 56 Robbins Road, Arlington, MA 02476 (US). HOOVER, Susan, B. [US/US]; 5020 Thornbark Drive, Barrington, IL 60010 (US). MURPHY, Mary, C. [US/US]; 47 Hearthside Circle, Bedford, NH 03110 (US). LOPRESTI, Patrick, J. [US/US]; 61A Sixth Street, Cambridge, MA 02141 (US). FUCARILE, Lori, J. [US/US]; 53 Melrose Street, 2A, Melrose, MA 02176 (US).

(74) Agents: PISANO, Nicola, A. et al.; Fish & Neave, 1251 Avenue of the Americas, New York, NY 10020 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR LICENSING CONTENT



(57) Abstract

Described is a method and system for managing access to functionality provided by programs by embedding license information in the content (201) that needs that functionality. This facilitates the distribution of both revenue and non–revenue generating content (201) on the Internet. A store (406) of license information can be maintained that identifies a licensee and an associated level of functionality for that licensee. This information defines an access policy for the licensee that determines what access to the functionality is granted to the licensee. The content (201) can then use the functionality of the programs to the extent allowed by the access policy.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEM AND METHOD FOR LICENSING CONTENT

Field of the Invention

The present invention relates generally to systems and methods for controlling and monitoring use of computer programs and other digital content, and more specifically to systems and methods for embedded licensing information in the computer programs and other digital content and for controlling access thereto in a distributed computer environment.

Background of the Invention

The Internet, and particularly the World Wide Web ("web" or "WWW") within the Internet, has become a popular vehicle for widespread access to content of nearly arbitrary kinds, including formatted text, graphics, audio, video, and interactive programmed content. Associated with various types of content are conventions, formats, or languages for its representation as strings of bits or characters when stored as files or communicated over a network. Although many content representations are known, currently popular representations include HTML for formatted text, JPEG for images, and JAVA for programs. Additional content representations are continually being developed. The CURL language, developed at the Massachusetts Institute of Technology (*Curl: A Gentle Slope Language for the Web*, M. Hostetter, D. Kranz, C. Seed, C. Terman, S. Ward, MIT Laboratory for Computer Science World Wide Web Journal, Volume 11. II, Issue 2, Spring 1997), is designed to represent a variety of different types of content within a single language.

Typically, content must be processed before it is presented to a user. Such processing requires software on the user's computer (the "client") which (a) fetches the content, represented in some identifiable language, from either a local storage medium (i.e., CD-ROM or hard disk) or remote computer (the "server") and (b) "executes," "interprets," or "processes" the fetched content appropriately. It should be noted that executing, operating on, and interpreting are all forms of processing content and used interchangeably herein.

Generally, an appropriate type of processing is readily apparent from the type of content being processed. This may be determined from a filename associated with the content, or from the content itself. For example, a filename ending in ".jpg" customarily denotes that the content is a JPEG image and that appropriate processing may include displaying the image on a computer monitor for viewing by the user. Analogously, content transmitted with a "midi" MIME type would be understood to be audio content to be played for a user to hear. Note, however, that other types of processing may be performed, such as displaying audio content using a suitable notation system so that it may be edited.

On the web, the acts of fetching and processing content are typically performed by a client application called a "browser," such as Microsoft Internet Explorer and Netscape Communicator, available from Microsoft Corporation, of Redmond, WA, and Netscape Communications Corporation, of Mountain View, CA, respectively. These browsers include the capability of processing several common types of content including HTML text, JPEG images, and Java programs, among others. These

browsers also accommodate new types of content, such as the CURL language, through loadable software modules commonly referred to as plug-ins.

When a browser attempts to fetch or load a non-native type of content, the browser loads a plug-in associated with the content type. Control is then passed to the plug-in to process the content. If an appropriate plug-in is not available to the browser on the client computer, an appropriate plug-in may be downloaded obtained from another source, typically as server on the Internet.

For example, when a browser fetches content comprising a CURL program, the browser locates and loads a corresponding CURL "plug-in" designed to process the CURL language. This may comprise compiling the CURL source code and executing the resulting code. Thereafter, when subsequent CURL language content is fetched by the browser, control is passed to the CURL language plug-in without having to reload the plug-in.

One reason for the success of the web is its promotion of unencumbered communications. An unambiguous identifier of information (e.g., a Uniform Resource Locator or URL) is typically all that is needed to enable access to content, a feature that has promoted the rapid evolution of the web into a rich nexus of interrelated content. While unconstrained access to information is valuable to the growth of the web and the electronic economy, it runs counter to conventional licensing models. Free and easy access to useful web content removes the motivation of those who enjoy its advantages to properly license and contribute revenue to the content supplier because it is difficult to control access and thus charge for information and services. Consequently, typical for-profit publishers of valuable content, such as computer

programs, research reports, timely sports scores, stock market information, and the like may forego strict licensing and encourage revenues by promoting an artificial scarcity of the commodity they provide. In those instances that a provider supplies coded content as well as the client software, or plug-in, needed to interpret the content, access restrictions may be applied to the content itself, to the plug-in, or to both.

Typical access restriction measures in current use tend to impede the open unencumbered communications that are the hallmark of the web. Such restrictions include passwords keyed to the user or passwords keyed to a hardware identification number attached to a specific computer and thus used to unlock the software or content. Password mechanisms can effectively limit the use of the software to a specific individual and/or specific machine but at the same time impede the web's unencumbered communications and further fail to scale for widespread use. Every user or provider of content must be provided a password to unlock the software or content.

Other mechanisms include removable hardware devices, commonly referred to as a "dongle," containing a unique license key. These devices are adapted to connect to a general-purpose communications port or specific interface of a client computer. A routine within a software program looks for a specific code contained in the removable device each time the software is activated to determine whether the software may be run. Again, such a mechanism can effectively limit the use of the software to a specific individual and/or specific machine, but, because of the need to distribute hardware devices, fails to scale for widespread use and thus impedes unencumbered communications.

Networked computer environments provide further licensing challenges. The terms of the traditional single-computer software license might not properly cover usage of software on a network or may unintentionally allow such a usage without additional compensation to the licensor. For instance, what form of license should be used when a user at one client uses software running on a second client. One solution is to grant a license to use the software on any computer connected to the network. This is known as an unlimited site license. Alternatively, a license, called a limited site license, may be required for each machine on which the software is to be run. While this may increases fees the licensor may collect, it may also require the licensee to pay for more licenses than are actually needed. In addition, this mechanism requires cooperation from the licensee to contact the licensor as additional nodes are added to the network. Additional non-automated mechanisms may be required monitor program usage and verify license compliance. For example, a licensor may require a licensee keep accurate records of usage, and may demand the right to periodically inspect the licensee's site to audit program usage; clearly, an inefficient and intrusive process.

In a situation where a licensor develops tools used by a licensee to create programs and content for an end-user, a further tension exists. The licensor may want an accurate accounting of the use of the licensed tools, while the ultimate end-user wants unencumbered access to the content without passwords and locks. The licensee is thus caught between providing the licensor an accurate accounting for determining a license fee while at the same time providing easy access to the end-user. Current

mechanisms that satisfy the licensor have an unfavorable impact on the end-user, and visa-versa.

Moreover, the developer of content might wish to encourage multiple levels of licensing to the end-user. The levels of licensing may run from free access to the product for certain uses, such as for evaluation or non-commercial use, to fee-based access for commercial use, wherein a fee is charged based on usage of the licensed product.

Conventional access restrictions constitute an impediment to free access by users, since such users must typically "register" with the access control mechanism or deal with an electronic bureaucracy. They are also an impediment to fee-based access because of the overhead required to monitor licensing compliance. Thus, present licensing technologies do not provide a broad mechanism capable of covering such divergent licensing goals.

Summary of the Invention

The invention provides a new and improved method and system for licensing content and for licensing access to functions provided by a plug-in, application, applet, or other software used to process the content. The invention, therefore, facilitates the distribution of both revenue and non-revenue generating content on the Internet.

This is achieved by embedding, within the content, licensing data referred to herein as a licensing form. The licensing form itself is related to the type of license under which the content is made available. Software designed to process the content

includes routines for recognizing and validating a license form embedded in the content and limiting processing of or access to the content pursuant to the validity or terms of the license.

Therefore, it is an advantage of the invention to support a licensing mechanism by which the provider of software utilized in the interpretation of web content may license the use of that software for commercial purposes without controlling access to either the software or the content.

Further advantages of the invention include the ability to allow noncommercial or other specific use of the software, as well as anonymous use of the software.

Additional advantages of the invention allow the licensor to monitor licensed usage, to revoke the license as desired, and to charge fees based upon the usage of the software by the licensed content.

In addition, it is also an advantage of the invention to utilize a plain-text assertion embedded in the content stating that the content is for non-commercial use.

Further advantages of this invention provide for multiple licensors, each administering separate licenses pertaining to interdependent content modules.

Other advantages of the invention will become apparent to one of ordinary skill in the art in view of the figures and detailed description of the embodiments of the invention.

Brief Description of the Figures

The foregoing advantages of the invention will be more readily understood by reference to the following description taken with the accompanying, drawings in which:

FIG. 1 is a block diagram illustrating a computer system with a plug-in or application for receiving licensed content according to the invention;

FIG. 2 is a block diagram illustrating a computer system with a plug-in or application for receiving licensed content and a server for providing such content according to the invention;

FIG. 3 is a block diagram illustrating the composition of the {License ...} form according to the invention;

FIG. 4 is a block diagram illustrating interactions between a computer system with a plug-in or application for receiving licensed content, a server for providing the licensed content, and a license server according to the invention;

FIG. 5 is a simplified flow chart illustrating steps performed by a plug-in when determining the type of license embedded within content being processed in accordance with the principles of the present invention;

FIG. 6 is a simplified flow chart illustrating steps performed by a plug-in of the invention when validating an implicit license;

FIG. 7 is a simplified flow chart illustrating steps performed by a plug-in of the present invention when validating an implicit license containing a content authenticator;

FIG. 8 is a simplified flow chart illustrating steps performed by a plug-in of the invention when validating an implicit license containing a content authenticator and further determining an access policy;

FIG. 9 is a simplified flow chart of steps performed by a plug-in of the invention when validating an explicit license containing a content authenticator, and further incorporating the use of a cache of valid license results and grace period;

FIG. 10 is a simplified flow chart illustrating a portion of the steps performed by a server in accordance with the principles of the present invention when validating an explicit license containing a content authenticator, and further incorporating the use of a cache of valid license results and grace period;

FIG. 11 is a block diagram of interactions between a computer system with a plug-in or application for receiving licensed content, a server for providing licensed content, and two license servers according to the invention; and

FIG. 12 is a block diagram illustrating environments created during multi-level licensing.

Detailed Description of the Preferred Embodiments

The following description is illustrative of a preferred embodiment of the present invention, and reflects concrete choices for purposes of illustration only. Many alternative embodiments of this invention will be apparent to those skilled in the art.

The invention facilitates the widespread use and distribution of computer programs yet provides developers of those computer programs a mechanism for garnering a financial return. This is accomplished by retaining control over the operations of the programs so as to control access to content requiring the programs. Thus, for example, non-commercial users of the programs may have free access (with possible limitations) while commercial users must pay a license fee for access to the operation of the computer programs.

In a first embodiment, the invention comprises a freely distributed software module, program, or plug-in that runs on a computer to process digital content. While processing the content, the module searches for data comprising a special licensing form, as described below. The licensing form must be present in the content for processing of the digital content to be authorized.

The module interprets the licensing form and authorizes processing of content based upon information in the licensing form, such as the nature of the content, e.g., non-commercial or commercial. The module has a built-in understanding of a non-commercial licensing form and can grant free access to the computer programs. The module also recognizes licensing forms of a commercial nature and can impose

further licensing validation steps (to ensure payment of a license fee for commercial use) prior to accessing the computer programs. Thus, the module has a built-in understanding of different types of license forms and can grant immediate access to the computer programs by non-commercial content while preserving control over access to the computer programs by commercial content. This encourages the development of content by allowing free access to the computer programs for non-commercial use while at the same time providing a mechanism for developers of the computer program to receive a financial return by requiring a license fee for commercial use.

FIG. 1(a) and FIG. 1(b) are block diagrams of illustrative embodiments of a computer system implementing the invention. The system utilizes computer 100 running operating system 101. Computer 100 can be a conventional personal computer such as those sold by Compaq Computer Corporation, IBM, and Apple. Computer 100 may also be another device such as a game console, set-top box, or Internet appliance. Operating system 101 is any operating system compatible with the underlying computer, such as Microsoft Windows 98, Microsoft Windows NT, Linux, Apple OS, and the like.

With reference to FIG. 1(a), a first embodiment of the invention uses browser 102 as a substrate for plug-in 103. FIG. 1(b) illustrates an alternative embodiment of the invention where application 104 is provided to instead of browser 102 and plug-in 103. In the embodiment of FIG. 1(b), the mechanism for controlling access to a computer program is included with in the computer program itself; however, one of

ordinary skill in the art would appreciate that the access controlling portion may be separated from the underlying computer programs.

The browser–plug-in combination, or equivalently application 104, is designed to interpret, execute, or otherwise process content received by computer 100. In particular, they are designed to recognize the licensing mechanism as taught by the invention and process the content accordingly. While the remaining description refers generally to plug-in 103 operating in conjunction with browser 102, it is understood by one of ordinary skill in the art that application 104 may be used interchangeably therewith.

Turning now to FIG. 2, the general operation of plug-in 103 is described as it processes content 201 received from server 200 using standard communications techniques. Content 201 is coded in a language or representation (such as the CURL Language) that plug-in 103 can ultimately process and may comprise a single component or a set of related components, such as web pages or program modules. In accordance with the principles of the present invention, multiple components may be licensed together as a whole, as well as individually, in groups, or some combination thereof.

Under the invention, content that is licensed to operate under plug-in 103 is distinguished by a special licensing form, or construct, within the content. Preferably, the licensing form is encoded so as to be compatible with the representation used to encode the content itself.

For simplicity, the invention is described herein using non-encrypted plain text CURL language source code. CURL language constructs are typically delimited by curly brackets. For example, the square of 9 may be calculated using a CURL construct such as:

```
{ square 9.0 }
```

Accordingly a suitable licensing form using CURL syntactic conventions may comprise a CURL language construct such as:

```
{Licensed ...}
```

wherein the ellipses denote further parameters to be described herein below.

A variety of alternative representations (e.g., HTML, Java, XML), data forms (e.g., text, graphics, executable code), and encodings (e.g., text, binary, or encrypted encodings) may be used with the present invention. Extending such representations and encodings to support the present invention may use any convenient alternative syntax. Typically such extension is suggested by existing syntactic conventions of the language or representation being extended in a manner analogous to the way license form used in CURL source code extends the CURL language.

A corresponding license form in HTML or XML, for example, might use the syntax

```
<LICENSED> ... </LICENSED>
```

while in C or Java it might take the form of a language construct such as

```
Licensed ...;
```

Similarly, the extension of binary representations (such as compiled class files, ActiveX controls, and dynamically loaded libraries) accommodate the existing structure of the representation so that the added information is unambiguously distinguishable from the remaining content. Such extensions will be apparent to those skilled in the art.

Whatever the representation, plug-in 103 is adapted to detect such appropriate {Licensed ...} form in every file delivered to it for execution, and to refuse execution of any file missing a valid {Licensed ...} form. Such refusal may be reported to the user of the client machine, the source of the content, or the plug-in provider. Alternatively, a file lacking a valid {Licensed ...} form may be processed in accordance with a default or limited behavior. For example, plug-in functionality may be restricted or a plug-in may be operational only for a specified period of time. In addition, the invention may provide for different licensing schemes by including different or additional information in the {Licensed ...} form.

In accordance with the principles of the present invention, plug-in 103 defers useful processing of content 201 until it has encountered a {Licensed ...} form contained within content 201. Typically, such {Licensed ...} forms will be at the beginning of each content file so that remaining content may be processed incrementally after validation of the {Licensed ...} form. Alternatively, some portion of the content may be processed in parallel with detecting and validating the {Licensed ...} form.

The {Licensed ...} form is used by plug-in 103 to control access to content 201, to plug-in 103 itself, or both. In other words, {Licensed ...} form 203 is processed by plug-in 103 to determine the extent to which content 201 is to be processed. Depending upon information in {Licensed ...} form 203, the final processing of the content 201 and the functionality of the plug-in 103 can be controlled. Thus, when content 201 is interpreted by client 100, its contents are delivered to plug-in 103, {Licensed ...} form 203 is extracted and processed, and plug-in 103 executes or interprets content 201 to the extent allowed by {Licensed ...} form 203.

FIG. 3 is an illustrative embodiment of a license for in accordance with the principles of the present invention. {Licensed ...} form 300 may contain a variety of information, including:

Identity of License 301 - this information identifies the licensee. In one embodiment of the invention, this information comprises plain text but in other embodiments it may be encrypted. The licensee is the party providing content 201 to be processed by plug-in 103, e.g., the content creator.

Legal Assertion 302 - this identifies the legal form of the license. In a first type of license form this field indicates that the licensed content may be distributed without payment of a royalty under a non-commercial use license. In other types of license forms this field specifies a type of license. The form of the assertion should be appropriate for the intended use of the content, and is

preferably a plain text string containing a legal assertion as to the nature of the license. It is anticipated that the assertion is appropriate for the licensee's jurisdiction and conforms to the laws governing the licensee and so that it may be used to judicially enforce the licensor's rights.

Encoded/Encrypted Data 303 - This portion of licensing form 300 includes a variety of data and may be provided in encrypted form. This data may be encoded as a sequence of plain-text characters using known techniques such as uuencode. In the exemplary licensing form for use with CURL source code, the data is quoted so that it becomes a syntactically legal element of the language (viz., a quoted string). Significant functions served by the encoded binary data are:

1. Secure identification of the licensee and the specific contractual arrangements pertaining, to the license;

2. Authentication of the content to detect tampering or modification of the contents and/or licensing form; and

3. Memorializing the content of a license applying to the content.

The following are examples of data that may be encoded/encrypted:

Licensee 304 - Represents the identity of the license. This can be used to confirm the identity of a licensee and may be a text string or numeric identifier. For example, a particular contract pertaining to the licensor/licensee in question

may be assigned a unique identifier C. Reliable encoding of identifier C in {Licensed...} form 300 then serves to identify the licensee and contractual arrangements - the first of the noted functions.

Time Out Period 305 - A period in which the license is valid. Can be used as an optimization of the verification phase of the license.

Grace Period 306 - Period of time to allow use of the content without verification of the license. Can be used as an optimization of the verification phase of the license.

Licensed Modules 307 - Extent of the license. This data can be used as part of a mechanism to control access to the plug-in functionality on a module-by-module basis. This enables licensing of only a portion of the functionality of plug-in 103.

Verification Site(s) 308 - One or more network locations where an authorization and verification server for the content license may be found as further explained herein below.

Unique ID 309 - An identifier used to identify the end user, the party actually downloading the content. This would allow a mechanism to track usage by the actual end-user of the content.

Authentication Code 310 - Information to verify that the code attached to the license has not been corrupted or tampered with. Implemented via a Message

Authentication Code or MAC computed from the remainder of the file, using known techniques (such as cipher-block chaining), thus providing the second noted function.

License Authentication Code 311 - Information to identify the terms under which the digital content is licensed. This may comprise for example a cryptographic hash or MAC computed from the text of the license which may be used as evidence of the terms of the license. Optionally the hash may be digitally time stamped and signed by the parties involved.

One of ordinary skill in the art would understand that additional data fields might be provided within {Licensed ...} form 300.

In accordance with the principles of the present invention, {Licensed ...} form 300 enables a licensor to implement different types of licensing schemes, or policies, for content. For example, a plug-in creator may choose a licensing policy that encourages use of a plug-in for certain purposes, such as for non-commercial use or for product evaluation. To support such a licensing policy plug-in 103 may accept certain variants of {Licensed ...} form 300 without further validation. This type of {Licensed ...} form is termed an implicit *license* because the plug-in has sufficient built-in information to grant access.

An exemplary implicit {Licensed ...} form may comprise:

```
{Licensed for non-commercial use under the terms of
Curl Corporation contract number C00001.}
```

This form contains only a plain text, legally binding assertion appropriate for the intended use--that the content is for non-commercial use for which Curl Corporation requires no further licensing arrangements. In this approach, the licensor depends upon the plain-text legal assertion within the {Licensed ...} form itself stating clearly the non-commercial nature of the content license. The legal assertion is readily viewable by anyone with viewing the content, and clearly states the intended non-commercial nature of the content. Alternatively, when the content is processed a pop-up dialog box or splash screen may be displayed informing the user of its non-commercial nature.

A plug-in creator may instead choose a licensing policy that encourages use of the plug-in for certain controlled distribution purposes, e.g., commercial use or secure access. Thus, in another embodiment of the present invention, plug-in 103 is adapted to require further validation of certain variants of the {Licensed ...} form 300. This type of {Licensed ...} form is termed an *explicit license* because plug-in 103 has insufficient built-in information to grant access, and must resorting to external information to validate the license form.

An exemplary explicit {Licensed ...} form may comprise:

```
{Licensed by XYZ Computer Corporation
    |"101010111001 . . . 00101010010000l001001"|.}
```

Wherein `XYZ Computer Corporation` identifies the licensee and the sequence of binary digits represents additional encrypted information 303 of FIG. 3. This

{Licensed ...} form may be distinguished from the prior form by the content of the license form itself. For example, the second word or term of the two exemplary {Licensed ...} forms differ, i.e., 'for' and 'by'. Clearly, other means of distinguishing the {Licensed ...} forms will be apparent to one skilled in the art.

When an explicit license form is encountered, encrypted information 303 is decrypted by plug-in 103 and additional operations are performed by plug-in 103 to verify the license. Plug-in 103 may also verify that the content has not been edited or otherwise tampered with. The current status of the license referenced by the {License...} form may be verified by a brief exchange of messages with a license server maintained by a Licensor.

FIG. 4 illustrates the flow of information between client 100, license server 403, and content server 200 in accordance with the principles of the present invention. Client 100 sends requests 401 for content 201 over a network, such as the Internet, to content server 200. Content server 200 sends response 402 back to client 100 containing requested content 201. Plug-in 103 then scans content 201 for a {Licensed ...} form prior to actual execution, interpretation, or processing of content 201. If a {Licensed ...} form is found, it is interpreted and verified by plug-in 103 in a manner dependant upon the contents of the {Licensed ...} form.

For explicitly licensed content, {Licensed ...} form 300 contains at least a some encoded information as shown by encoded information 303 of FIG. 3. To verify this form of license plug-in 103 creates license validation request 404 based in part

{Licensed ...} form 300 and encoded information 303 contained therein. License validation request 404 is sent to authorization server 403.

Authorization server 403 may be the same server as content server 200, but in the present example, authorization server 403 is a separate computer. Authorization server 403 receives license validation request 404 from client 100and generates response 405 dependent upon information in license validation request 404 and the contents of license database 406 which is discussed further below.

The flow charts of FIGS. 5-10 illustrate an exemplary sequence of steps performed by embodiments of plug-in 103 when validating a {Licensed ...} form. Referring first to FIG. 5, plug-in 103 performs an initial sequence of steps to retrieve and determine a type of license designated by the {Licensed ...} form. At step 500, plug-in 103 is in an initial state ready to retrieve content located on a server. In step 501, plug-in 103 retrieves the content from the server. Retrieval may be initiated by the user or under operation of previously loaded content. For example, a user may click on a hyper-text link, or previously loaded content may reference additional content to be retrieved. In a typical embodiment of the invention, the actual transport mechanism for retrieving content from the server is provided by the underlying browser and operating system, but it is anticipated that in other embodiments of the invention the plug-in might include the retrieval mechanism. For instance, a program for controlling a stand alone Internet appliance or consumer set-top box may contain code implementing a retrieval mechanism directly.

At step 502, the plug-in scans the content for the {Licensed ...} form. Typically, the form will be at the top of the file to minimize the time required for scanning the content. If there is no {Licensed ...} form within the content, it is signaled as an error, step 504, and the plug-in exits this routine, step 505.

Assuming at least one {Licensed ...} form is found, it is decoded to determine the type of license, step 506. The plug-in branches at step 507 depending upon the type of {Licensed ...} form found in the content. As described herein, there are two basic types of license, implicit and explicit; however, it is anticipated that other forms of license may be created. If the license is implicit, it branches to steps starting at step 508, otherwise it branches to steps 509.

Turning now to FIG. 6, if the {Licensed ...} form specifies an implicit license, the validity of the license is now checked. At step 510, the implicit license is checked, for example, by simply matching a text string in the {Licensed...} form containing the legal assertion to a text string retained by the plug-in. Alternative methods of checking an implied license may include comparing the results of a hash or other function applied to the legal assertion to a value stored by the plug-in.

At step 511, if the license check fails, the plug-in executes any necessary error operations, step 512, and exits the function at step 513. If the license check passes, the plug-in operates on the content to the extent allowed by the license terms and the functionality provided by the plug-in. Once the plug-in is allowed to operate on the content, this function is exited at step 515.

FIG. 7 illustrates an alternative embodiment of the invention for operating on an implicit license in which the content is secured by a MAC. At step 516, the implicit license is checked as described above. If the match fails at step 517, the plug-in executes the necessary error operations, step 518, and exits this function at step 519. If, however, the license check is successful, the plug-in computes a MAC from the contents and compares it with the MAC found within the {Licensed ...} form, step 520. If the comparison fails at step 521, indicating that the content has been corrupted and should not be operated upon, then an error condition is signaled at step 522, and the function exits at step 523. If the comparison succeeds at step 521, then the plug-in operates on the content to the extent allowed, step 525, and the function exits at step 515. One of skill in the art will understand that various other functions may be provided by the {Licensed ...} form in an analogous manner.

Referring now to FIG. 8, an additional extension of the preferred embodiment is described. The steps illustrated in FIG. 8 are the same as those in FIG. 7, with the addition of step 524, wherein the plug-in determines an access policy appropriate for the license. The access policy may limit the plug-in to a set of permissible functionality. For example, a basic set of functions may be made available under a non-commercial use license, while at the same time withholding access to advanced features unless they are enabled under an explicit licensing regime, as described below.

FIG. 9 illustrates steps performed by the plug-in when it is determined that the license is an explicit license. In explicitly licensed content, the {Licensed ...} form will always have an encoded portion such as encoded portion 303 of FIG. 3, containing

additional information needed to validate the license. At step 550, the encoded information is decoded. Depending upon the encoding, a variety of levels of security surrounding access to the encoded information can be provided. For instance, the encoding comprise a simple binary encoding, or, alternatively, may cryptographic techniques using a public or private key infrastructure to both encrypt and encode the data.

At step 551, a MAC of the contents is computed and compared to MAC 310 of {Licensed...} form 300 to determine whether the contents have been corrupted or the {Licensed ...} form has been illegally generated or copied. If the computed MAC is not acceptable, step 553, an error condition is signaled, step 553, and the routine exits 554. If the computed MAC is authentic, at step 555 a cache is checked to see if a prior approved {Licensed ...} form applies to the content under examination. If a cache entry is found that applies to the content, tested at step 556, then the cache entry is updated at step 557 (to allow for aging of the entry via an access count, usage limit or other aging mechanism).

If there isn't a valid cache entry, then a license server needs to be contacted at step 560. Identification of the license server can be built into the plug-in, specified by the encoded information (303 of FIG. 3), specified by another server, or built into the content. The plug-in attempts to contact the license server and pass the necessary information from the plug-in to the license server as required for validation of the license. The necessary information can include one or all of the following: licensee identity, content identity, user identity, machine identity, contract identity, use count,

etc. Preferably, the information is passed to the license server via a secure communications link as known in the art to protect the communications from interception.

One important aspect of the invention is that some of encoded information 303 of FIG. 3 may be stored on the license server and a reference to the data stored in the {Licensed ...} form. For instance, the license server may specify a timeout period for a cache entry associated with specific content, an additional verification site, another module that the license applies to, or a grace period applicable to the content, etc. Thus, by storing some information on the license server, the values can change across all accesses of the content without requiring a change to the {Licensed ...} form in the content. In addition, with a user or machine identification, a customized access policy can be created on a per machine or per user basis. Finally, even if the encoded information (303 of FIG. 3) includes certain information, that information can be overridden by information found on the license server.

If the license server does not respond, additional license servers may be queried if the location of more than one license server is provided in verification sites field 308 of FIG. 3. Otherwise, at step 561, a grace period may apply, step 562, even if there isn't a cache entry nor a response from the server. A grace period is an optimization to enable disconnected operations. For instance, content may be stored on the user's machine for later access. If the user is not connected to a network containing a license server, then without a cache entry from a previous access, operation of the content would be denied. A grace period allows the content to be used a certain number of

times, or for a certain time period, without requiring access to the license server. If the grace period is expired, step 563, an error is signaled, step 564, and the content is not operated upon and the routine exits, step 565. If the grace period applies, and is valid, then the content is operated to the extent allowed by the license, step 566, and the routine exits.

FIG. 10 illustrates the steps in contacting the server for one embodiment of the invention. If a server responds, we enter at step 568. The response from the license server will typically include additional information, as already noted. The information received is used to validate or invalidate the license. If the response invalidates the license, step 570, then an error is signaled, step 571, and the routine exits, step 572. If the response validates the license, then an entry is created in the cache for the content, step 573, and the content is operated on to the extent allowed under the license, step 574, and the routine exits, step 575.

FIGS. 5-10 are only meant as examples of various embodiments of the licensing mechanism. One of ordinary skill in the art can appreciate that additional extensions and modifications are possible. For instance, the information contained in the {Licensed ...} form can be augmented, the information stored on the server can be augmented, the encryption and encoding algorithms can be changed, the communications method can be varied and all fall within the scope of the invention.

Extensions

Although the user computer - authorization server exchange can be made quite brief, it does add a small overhead to each execution of the licensed content. As already shown, this overhead can be reduced by caching the server response on the client (e.g., within a "cookie" or similar client state), along with a specified timeout interval used to control its life. The cached result may then be applied to future retrieval of related content. When requested to execute a licensed file, the plug-in would check its local cache for related records affirming the validity of the {Licensed ...} form. Thus, the plug-in need only exchange messages with a license server if there were no such cache entry or if the cache entry has expired.

Caching prior license approvals also reduces the number of license validation requests that must be sent to a license server. The number of license validation requests may be reduced even further by allowing cache entries that apply to multiple {Licensed ...} forms. For example, a cache entry may validate all {Licensed ...} forms having a specified content, naming a specified licensee, or originating from a specific domain are valid.

Various implementations of the timeout mechanism are possible. For instance, the actual timeout interval for the cached response might be constant (built into the plug-in), specified within the {Licensed ...} form, or specified by the Licensor's server in its response to the validation request. The timeout interval may be based upon passage of time, number or accesses or a combination thereof.

In addition, the relationship of the cached server response with future content retrieved by the user can also be configured. For instance, if the content is organized in a hierarchy, the cached response for a particular node can apply to that node and all nodes lower in the hierarchy or a limited number of lower nodes. Thus, a cached response for a web page may apply to all web pages within the same domain. Alternatively, the {Licensed ...} form or the response from the server can specify the content for which it applies. Thus, the cached response is flexible and can be adapted to support the licensee's particular content.

An alternative to the validation steps illustrated in FIGS. 5-10 above is to modify the contact with the license server to take place in the background. This would allow "lazy authentication" of the license, where the license is validated only as needed and then in parallel with operation on the content. From the user perspective, this could eliminate or diminish any delays caused by the interactions between user computer and the license computer prior to validation and thus operation of the content.

To save client storage and minimize the size of a {Licensed ...} form, a portion of it may be digested via a cryptographic hash function such as MD5, SHA1, or other hash function (*Applied Cryptography*, Second Edition, By Bruce Schneier, John Wiley & Sons, 1996) into a smaller form to be compared against stored values. Alternatively, the plug-in might deal with all {Licensed ...} forms by contacting s license server, sending a validation request containing, a short cryptographic hash of the form, and then caching the result to avoid the need for future validations. This

approach allows the set of implicit {Licensed ...} forms to grow dynamically, under control of the Licensor via the license server. In addition, the use of a cache timeout enables a licensor to revoke licenses, since cached validations will eventually expire.

Encoded information 303 of FIG. 3 of {Licensed ...} form 300 may be constructed from C, the MAC of the file contents, and other data in a number of ways. An exemplary calculation of encoded information 303 is shown as follows:

```
Encode(
   Encrypt(
      Append(
        Hash(Append(C,MAC(remainder of file))),
        C,
        MAC(remainder of file)
      ),
      SecretKey
   )
)
```

wherein `Append` simply concatenates the bit strings representing its arguments, `Hash` computes a noninvertible cryptographic hash of its arguments, `Encrypt(bits, key)` is a standard encryption scheme such as DES, and `Encode` is a character encoding function such as encode. `SecretKey` is a symmetric encryption key known only to the licensor. The hashing and encryption steps discourage attempts to separate the C and MAC components of the encoded binary data and recombine them in forged {Licensed ...} forms.

Since occasional re-validation of each explicit (and some versions of implicit) license is required, a license server may collect and store in a database data corresponding to the licenses. Such license records may me analyzed statistically to

yield a rough model of the active user base served by each license. The licensor might, for example, send weekly reports to each licensee relating the number of active users, and perhaps asking the licensee to renegotiate a license if the number exceeds the terms of the current license. Alternatively, a plug-in may include the capability of retrieving license records from the license server so the information may be processed by the user.

In addition, a licensor may specify cache timeout values, the scope of cache entries, and organize content in such a way that the licensor may generate reports on the sequence or flow of access through the content on a content server, and thus be able to identify those content areas that are over or under utilized.

In the embodiments of the invention described herein, the invention is designed so that substantive processing of the encoded binary data can happen entirely on the server freeing client software from the need to include sophisticated encryption routines. However, in accordance with the principles of the present invention, these and other functions may be relocated between the plug-in and the licensor's server as desired, using network communication with standard security measures as necessary.

License Generation

For a licensee to create content licensed for use with the plug-in, the licensee must be able to affix the cryptographic {Licensed ...} form to the created content. To this end, the licensor supplies the licensee with a certification tool, namely a program specific to the appropriate contract (and hence specific to that licensee) which takes as

input an unlicenced file and affixes the appropriate {Licensed ...} form to it. Preferably, the license tool is integrated into the programs and routines used to originally create the content, e.g., in the form of a dynamic load library or other module. Alternatively, the tool may be a stand-alone program.

Clearly, the certification tool is intended only for use by a particular licensee, and should not be distributed publicly. In the event the security of the tool becomes compromised, the licensor can react by revoking the license it produces, and issuing the licensee a new tool which implements a fresh (uncompromised) license. The tool deletes any earlier {Licensed ...} form as necessary, and generates a fresh copy based on current licensing terms and the remaining contents of the supplied source file. Alternatively, a {Licensed...} form may include a time stamp so that a license server may only validate license forms having a time stamp prior to when the tool was compromised.

The tool may be completely self-contained, relying on internal coding and encryption mechanism, or may operate in conjunction with a server maintained by a particular licensor. At its extreme, under the latter approach, a {Licensed ...} form can be created by transmitting the content file (or the MAC of the file) via a secure link to the license creation server. The license creation server would then generate the necessary information and send it back in response. An advantage of this approach is that the tool containing a licensor's cryptographic information is kept under the control of the licensor.

Multi-layer Licensed Software

The {Licensed ...} form can be extended to support multiple licensor-licensee relationships. For instance, the system can support a relationship as shown in FIG. 11. In this example, the plug-in can be licensed to a content provider to develop and deploy an application. In this case, the licensor is the plug-in creator and the licensee is the content creator/provider. In turn, the content provider can license the content to a user. In this situation the licensor is the content creator/provider and the licensee is the user.

The operations parallel the examples described previously, but are now extended to handle two licensor-licensee relationships. Client 100 sends request 601 for content 201 to content server 200. Content server 200 responds by sending answer 602 containing content 201. Plug-in 103 scans content 201 for {Licensed ...} forms, and in this case, encounters two license forms, such as

```
{Licensed by Alpha Company under contract A23}
{Licensed by Bravo Company under contract B99}
```

Alternatively, multiple license forms may be combined or nested.

One {Licensed ...} form relates to a license between a plug-in provider and a content provider. Plug-in 103 contacts 603 plug-in license server 403(a) to validate the related {Licensed ...} form. License server 403(a) retrieves the appropriate information from license database 406(a) and send respond 604 to plug-in 103.

The other {Licensed ...} form relates to a relationship between the content provider and the end user. Again, plug-in 103 sends message 605 to content license server 403(b) to validate the related {Licensed ...} form. License server 403(b) retrieves the appropriate information from license database 406(b) and sends respond 606 to plug-in 103.

Depending upon the responses received, 103 plug-in operates on the content accordingly. Each of the two {Licensed ...} forms may have different settings, such as different validation servers and different grace or cache timeout periods, thus enabling license interactions to be customized accordingly. For instance, the plug-in provider may provide a long cache timeout period for this particular licensee to minimize traffic to the plug-in provider's license server.

This general scheme is further extended in an additional embodiment of the present invention so that licensed modules may refer to each other. For instance, separately sourced content and software modules can refer to each other. In CURL, the form:

```
{require <file locator> ...}
```

specifies that execution of the module containing the {require ...} form depends on access to another module identified by the specified file locator, which is typically, a Universal Resource Locator ("URL"). When the plug-in encounters such a dependency, it fetches and imports the specified or "required" module. Both the required and requiring module contain {Licensed ...} forms, which may specify the

same or different licenses, licensors, and licensees. In the most general and most interesting case, the modules are from different developers.

For example, company B may provide which-performance high-performance Curl audio software, B.curl, of interest to other Curl developers, and company A might produce a Curl application A.curl which requires B.curl. Both A and B are licensees of Curl Corporation; but it is useful to provide additional support for A to become also a licensee of B. To this end, we provide the system provides a mechanism for allowing third-party software vendors to utilize aspects of our invention to administer their software licenses.

The form

```
{Licensor <name of licensor>
     <encoded binary data>)}
```

appearing within a file is used to establish that the containing content module is subject to licensing, restrictions in addition to those involving the plug-in itself. The name of licensor section is be plain text, and identifies the licensor (and optionally other pertinent details) in human-readable form. The encoded binary data includes, perhaps in encrypted form, a URL or other identifier of one or more servers that administer the indicated license. In the above example, the form

```
{Licensor B Corporation
     <encoded binary data>}
```

would appear within file B.curl to indicate that its use is subject to licensing restrictions imposed by B Corporation.

The {require ...} form, which indicates a dependency on external content, may include an optional {Licensed ...} form, properly nested as a clause within the {require ...} form. The {Licensed ...} clause has a structure and function analogous to the {Licensed ...} form already described, except that it is embedded within a {require ...} form and indicates that it pertains to a license from the Licensor declared in the required module.

In our example, file A.curl might include the form

```
{require "http://www.B.com/B.curl"
     {Licensed by A
          <encoded binary data>}
     ...}
```

wherein the `http://www.B.com/B.curl` is a URL or other reference to a module required by A.curl,, and wherein the {Licensed ...} clause contains within its encoded binary data the components previously described as necessary for B's license server to validate A's license status. On encountering one of the above forms within A.curl, the plug-in will first load B.curl appropriately satisfying all of its {require ...} forms and validating licenses as required. The plug-in will then validate A's use of B.curl as previously described, using data from the {Licensed by A ...} form in the {require ...} form together with the server identified in the {Licensed ...} form found within B.curl. B may provide implicit licenses, using B's server to validate each {Licensed ...} clause dynamically as previously described.

As part of A's licensing arrangement with B, A receives a certification tool from B which enables A to add the {Licensed ...} clause to the {require ...} form to

modules created by A. This tool operates in generally the same fashion as the certification tool previously described, except that it generates a clause rather than a top-level {Licensed ...} form.

Using this approach, B is able to license the use of B.curl to A, rather than simply licensing every use of B.curl to all of A's clients. As illustrated in FIG. 12, B can prevent the unlicenced D.curl from using B.curl even while A.curl is gainfully using B.curl on a given client. B.curl is dependent on the plug-in's ability to segregate modules of functionality such that only approved interactions and interdependence obtain. Many modern operating systems provide support for this, as do typical systems for mobile code such as Java (Sun Microsystems, Inc. Inc., Palo Alto, CA) and Safe-TCL *(The Safe-Tcl Security Model,* Jacob Y. Levy and Laurent Demailly, Sun Microsystems Laboratories; John K. Ousterhout and Brent B. Welch, Scriptics Inc. USENIX Annual Technical Conference (NO 98), 1998, June 15-19, 1998, New Orleans, Louisiana, USA). Moreover, B.curl can similarly require licensed software from additional vendors, under separately maintained licenses. In effect, the supplier of the plug-in can extend its licensing, advantages to arbitrary software vendors.

Other Embodiments

Many variants of the described embodiment will be apparent to those skilled in the art. For example, some or all of the content may be coded in binary, either as instructions for a real or virtual machine, or as an intermediate representation requiring further interpretation or translation.

Various additional data fields may be included in a {Licensed ...} form relating to additional features offered by a client plug-in. For example, a licensor may offer additional user authentication features such as requiring a plug-in to request identification and password information from a user, and optionally verify the with the licensee's server.

In additional embodiments of the invention, a plug-in may initially contain only those instructions needed for interpreting the {Licensed ...} form and for loading additional modules or plug-ins as may be necessary to process the content. In this case the plug-in or {Licensed ...} form will identify at least one server capable of providing the additional modules or plug-ins. These additional modules are downloaded and stored on the client to be used as needed. Optionally, the client may dynamically release unused modules depending upon the time of last access or such replacement metric. This provides for a very small plug-in as all other modules are subsequently downloaded from a server when required to process content and any {Licensed ...} form contained therein. Space occupied by released modules may then be reclaimed for reuse, thereby enabling the creation of a client with minimal module storage area.

The mechanism of this invention includes elements of authentication and certification that may play useful roles unrelated to the objects of the present invention, for example pursuant to privacy or security goals. In applications of the invention sharing these additional goals, the coding of information within {Licensed ...} forms may advantageously be combined with other information and mechanism serving these additional purposes. The combination of licensing and security

mechanism may be optimized in ways that will be obvious to those skilled in the art, eliminating redundant information and mechanism from the combined implementation.

The client execution vehicle need not be a browser plug-in as described here. It might be an application, applet, dynamically linked library (DLL), ActiveX control, or any other form of executable code.

What is claimed is:

1. A method of licensing digital content, the method comprising:

examining the digital content to identify data indicating that the content is subject to a license, wherein the data includes a license-specific portion;

determining a status of the license; and

processing the digital content subject to the status.

2. The method of claim 1 wherein examining the digital content comprises searching the digital content for a license statement.

3. The method of claim 1 wherein the data includes a human readable portion.

4. The method of claim 1 wherein the data includes an encrypted portion.

5. The method of claim 1 wherein determining a status of the license comprises determining the status of the license based on the data.

6. The method of claim 1 wherein determining a status of the license comprises requesting the status of the license from a server.

7. The method of claim 6 wherein requesting status from a server further comprises sending to the server a portion of the data.

8. The method of claim 7 wherein the data includes an encrypted portion and sending a portion of the data to the server comprises sending the encrypted portion.

9. The method of claim 6 further comprising caching a response from the server, wherein determining a status of the license comprises inspecting the cache for a response from the server and requesting the status of the license from the server responsive to the inspection of the cache.

10. The method of claim 9 wherein inspecting the cache comprises:

finding a cache entry corresponding to the data; and

determining the validity of the cache entry.

11. The method of claim 10 wherein determining the validity of the cache entry comprises determining that the cache entry has not expired.

12. The method of claim 6 wherein processing the digital content subject to the status comprises processing at least a portion of the digital content in parallel with determining the status.

13. The method of claim 1 wherein processing the digital content subject to the status comprises processing at least a portion of the digital content in parallel with determining the status.

14. The method of claim 1 wherein the data includes a portion for determining whether the digital content has been altered.

15. The method of claim 1 wherein processing the digital content comprises selectively applying first and second processing routines to the data subject to the determined status.

16. The method of claim 15 wherein selectively applying first and second processing routines to the data comprises applying the first processing routine to the data and not applying the second processing routine to the data.

17. The method of claim 1 wherein determining a status of the license comprises determining that the license is for non-commercial use.

18. The method of claim 17 further comprising indicating to a user that the content is licensed for non-commercial use.

19. A method of licensing digital content, the method comprising:

providing first software that embeds in the digital content data indicative of a license, the data including a license specific portion; and

providing second software that:

identifies the data in the digital content;

determines a status of the license indicated by the digital content; and

processes the digital content responsive to the status.

20. The method of claim 19 wherein identifying the data indicative of a license comprises searching the digital content for a license statement.

21. The method of claim 19 wherein the data includes a human readable portion.

22. The method of claim 19 wherein the data includes an encrypted portion.

23. The method of claim 19 wherein determining the status of the license comprises determining the status of the license based on the data.

24. The method of claim 19 wherein determining the status of the license comprises requesting the status of the license from a server.

25. The method of claim 24 wherein requesting the status from the server further comprises sending a portion of the data to the server.

26. The method of claim 25 wherein the data includes an encrypted portion and sending the portion of the data to the server comprises sending the encrypted portion.

27. The method of claim 24 further comprising caching a response from the server, wherein determining a status of the license comprises inspecting the cache for a response from the server and requesting the status of the license from the server responsive to the inspection of the cache.

28. The method of claim 27 wherein inspecting the cache comprises:

finding a cache entry corresponding to the license; and

determining the validity of the cache entry.

29. The method of claim 28 wherein determining the validity of the cache entry comprises determining that the cache entry has not expired.

30. The method of claim 24 wherein processing the digital content subject to the determined status comprises processing at least a portion of the digital content in parallel with determining the status.

31. The method of claim 19 wherein processing the digital content subject to the determined status comprises processing at least a portion of the digital content in parallel with determining the status.

32. The method of claim 19 wherein the data includes a portion for determining whether the digital content has been altered.

33. The method of claim 19 wherein processing the digital content comprises selectively applying first and second processing routines to the data subject to the determined status.

34. The method of claim 33 wherein selectively applying first and second processing routines to the data comprises applying the first processing routine to the data responsive to the determined status and not applying the second processing routine to the data responsive to the determined status.

35. The method of claim 19 wherein determining a status of the license comprises determining that the license is for non-commercial use of the content.

36. The method of claim 35 further comprising indicating to a user that the content is for non-commercial use.

37. Apparatus for licensing digital content, the apparatus comprising:

a general purpose computer; and

a memory, the memory including programmed instructions for:

examining the digital content to identify data indicating that the content is subject to a license, the data including a license-specific portion;

determining a status of the license; and

processing the digital content subject to the status.

38. The apparatus of claim 37 wherein examining the digital content comprises searching the digital content for a license statement.

39. The apparatus of claim 37 wherein the data includes a human readable portion and an encrypted portion.

40. The apparatus of claim 37 wherein determining the status of the license comprises determining the status of the license based on the data.

41. The apparatus of claim 1 wherein determining the status of the license comprises sending a portion of the data to a server and requesting the status of the license from the server based on the data .

42. The apparatus of claim 41 wherein the data includes an encrypted portion and sending a portion of the data to the server comprises sending the encrypted portion.

43. The apparatus of claim 41 further comprising caching a response from the server, wherein determining a status of the license comprises inspecting the cache

for a response from the server and requesting the status of the license from the server responsive to the inspection of the cache.

44. The apparatus of claim 43 wherein inspecting the cache comprises:

finding a cache entry corresponding to the data; and

determining that the cache entry has not expired.

45. The apparatus of claim 41 wherein processing the digital content subject to the status comprises processing at least a portion of the digital content in parallel with determining the status.

46. The apparatus of claim 37 wherein processing the digital content subject to the status comprises processing at least a portion of the digital content in parallel with determining the status.

47. The apparatus of claim 40 wherein the data includes a portion for determining whether the digital content has been altered.

48. The apparatus of claim 37 wherein processing the digital content comprises selectively applying first and second processing routines to the data subject to the determined status.

49. The apparatus of claim 48 wherein selectively applying first and second processing routines to the data comprises applying the first processing routine to the data and not applying the second processing routine to the data.

50. The apparatus of claim 37 wherein determining a status of the license comprises determining that the license is for non-commercial use and indicating to a user that the content is licensed for non-commercial use.

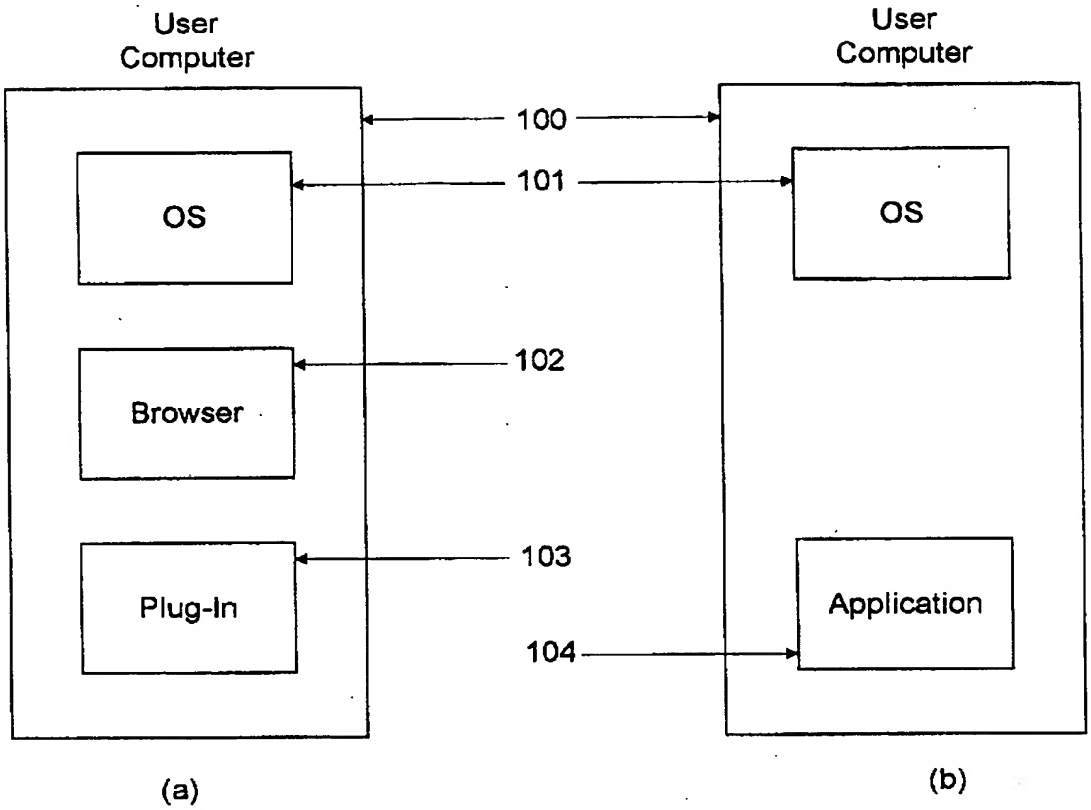

User Computer
User Computer
100
101
OS
OS
102
Browser
103
Plug-In
Application
104
(a)
(b)

FIG. 1

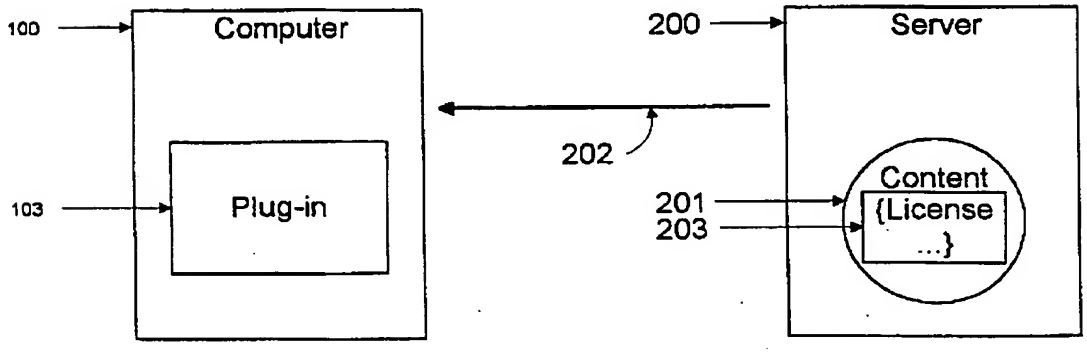

100
Computer
200
Server
202
103
Plug-in
201
203
Content
{License
...}

FIG. 2

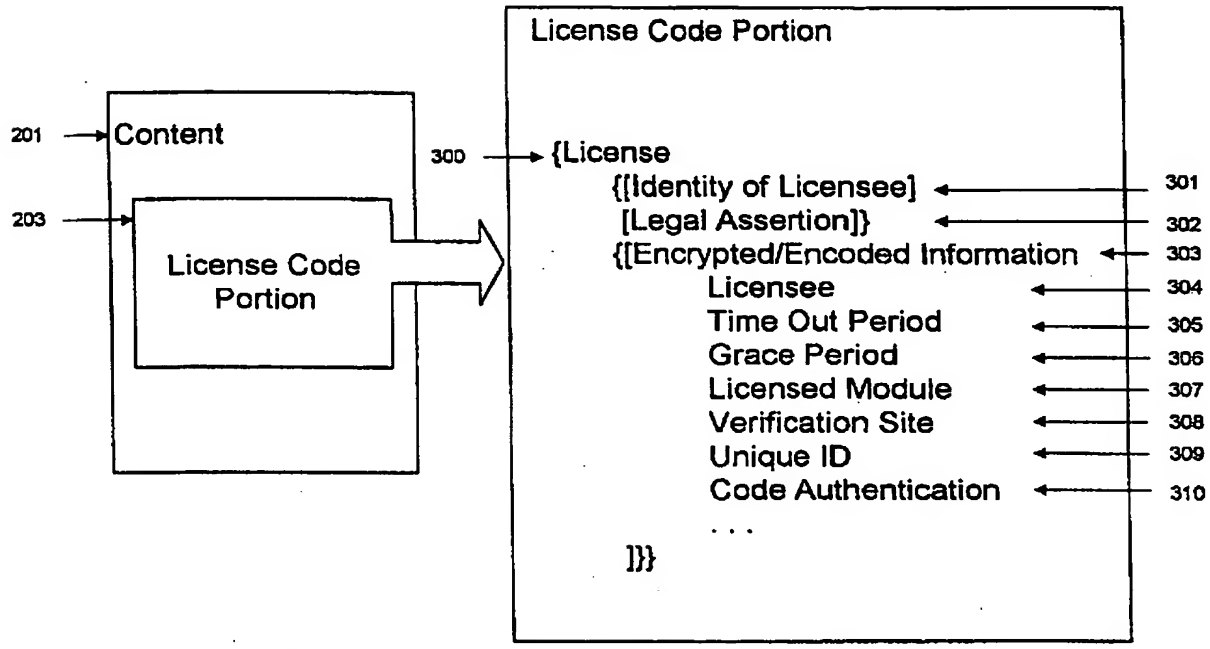

License Code Portion
201
Content
203
License Code Portion
300
{License
{[Identity of Licensee]
[Legal Assertion]}
{[Encrypted/Encoded Information
Licensee
Time Out Period
Grace Period
Licensed Module
Verification Site
Unique ID
Code Authentication
. . .
]}}
301
302
303
304
305
306
307
308
309
310

FIG. 3

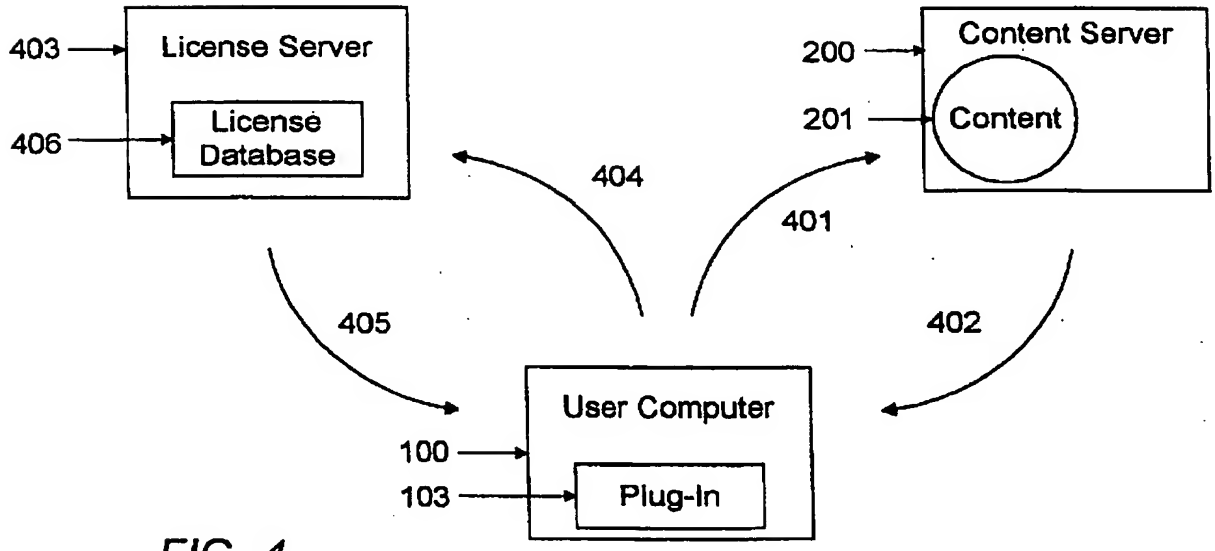

403
License Server
406
License Database
200
Content Server
201
Content
404
401
405
402
User Computer
100
103
Plug-In

FIG. 4

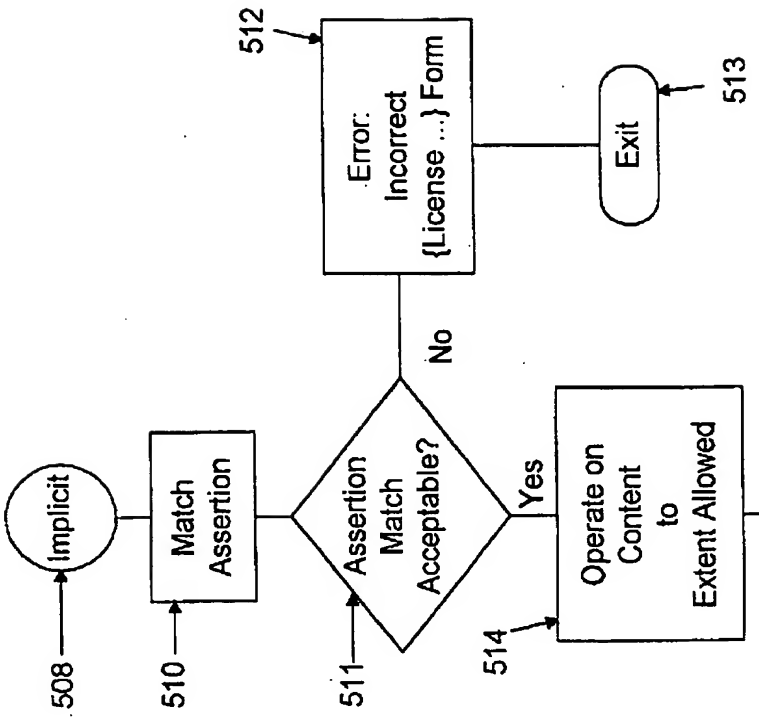

Implicit
508
Match Assertion
510
Assertion Match Acceptable?
511
No
Error: Incorrect {License ...} Form
512
Exit
513
Yes
Operate on Content to Extent Allowed
514
Exit
515

FIG. 6

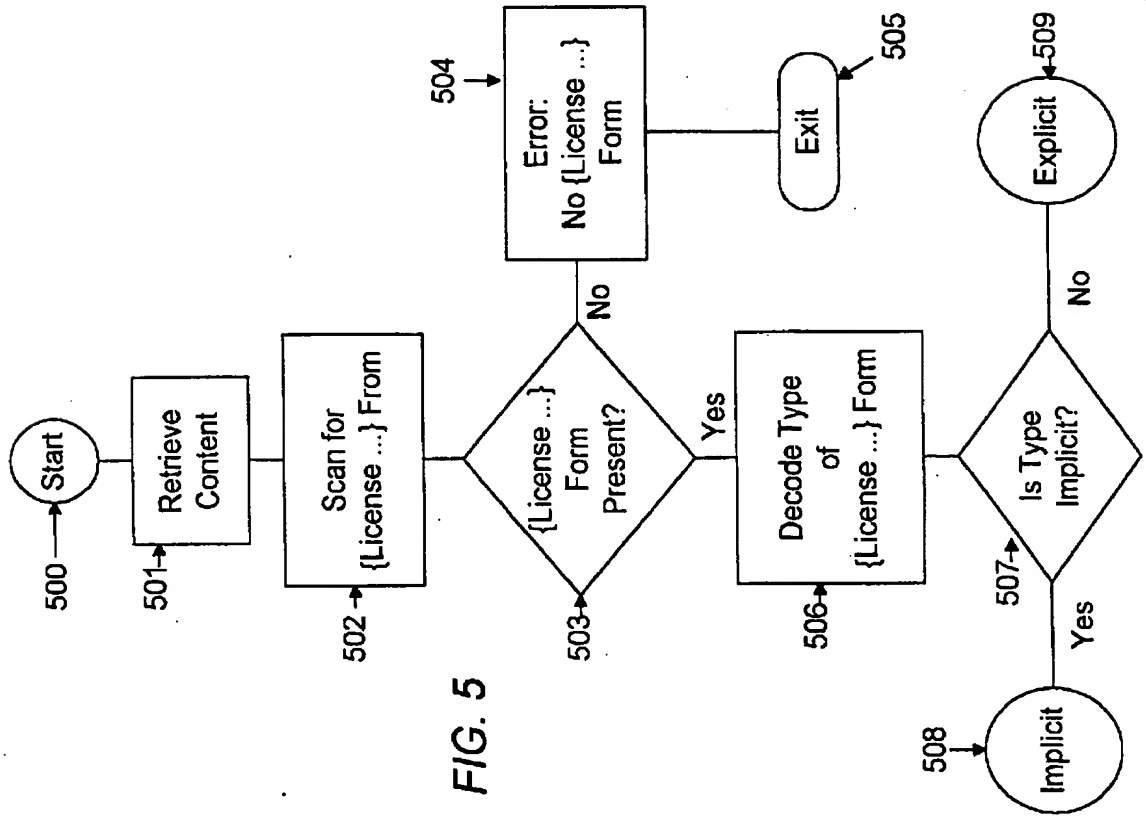

Start
500
Retrieve Content
501
Scan for {License ...} From
502
{License ...} Form Present?
503
No
Error: No {License ...} Form
504
Exit
505
Yes
Decode Type of {License ...} Form
506
Is Type Implicit?
507
No
Explicit
509
Yes
Implicit
508

FIG. 5

FIG. 7

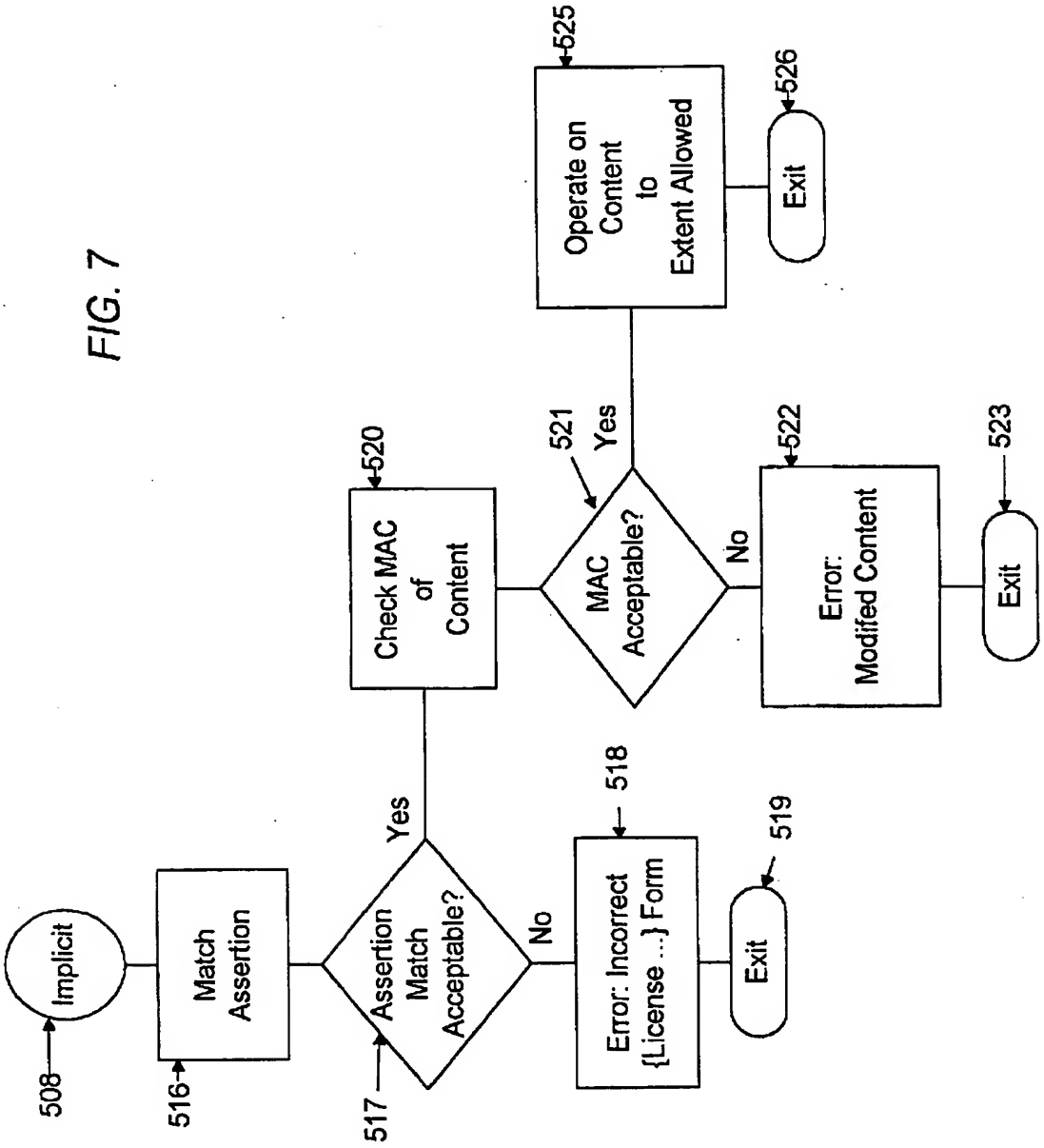
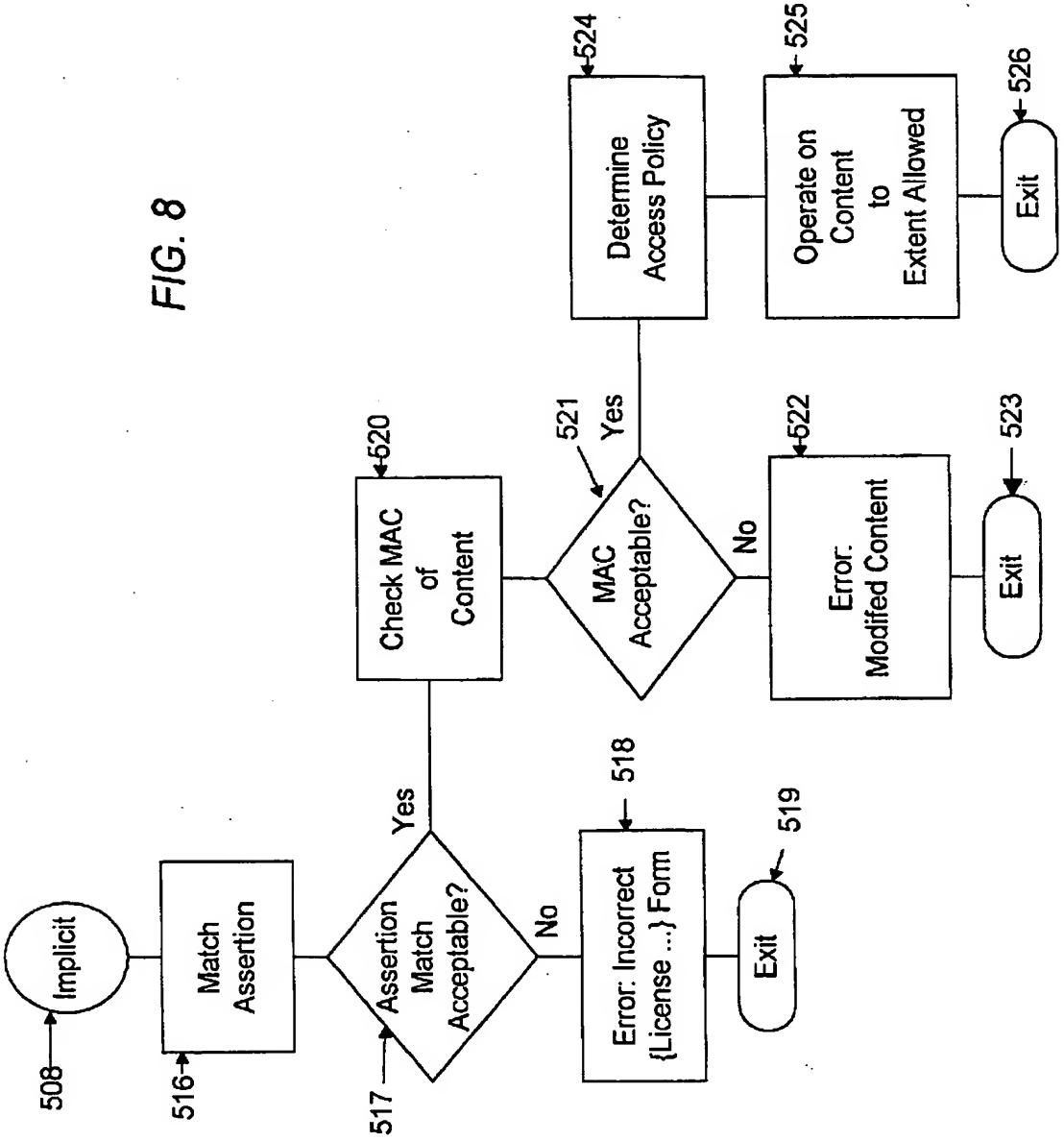

Implicit
508
Match Assertion
516
Assertion Match Acceptable?
517
Yes
No
Error: Incorrect {License ...} Form
518
Exit
519
Check MAC of Content
520
MAC Acceptable?
521
Yes
No
Error: Modifed Content
522
Exit
523
Operate on Content to Extent Allowed
525
Exit
526

FIG. 8


Implicit
508
Match Assertion
516
Assertion Match Acceptable?
517
Yes
No
Error: Incorrect {License ...} Form
518
Exit
519
Check MAC of Content
520
MAC Acceptable?
521
Yes
No
Error: Modifed Content
522
Exit
523
Determine Access Policy
524
Operate on Content to Extent Allowed
525
Exit
526

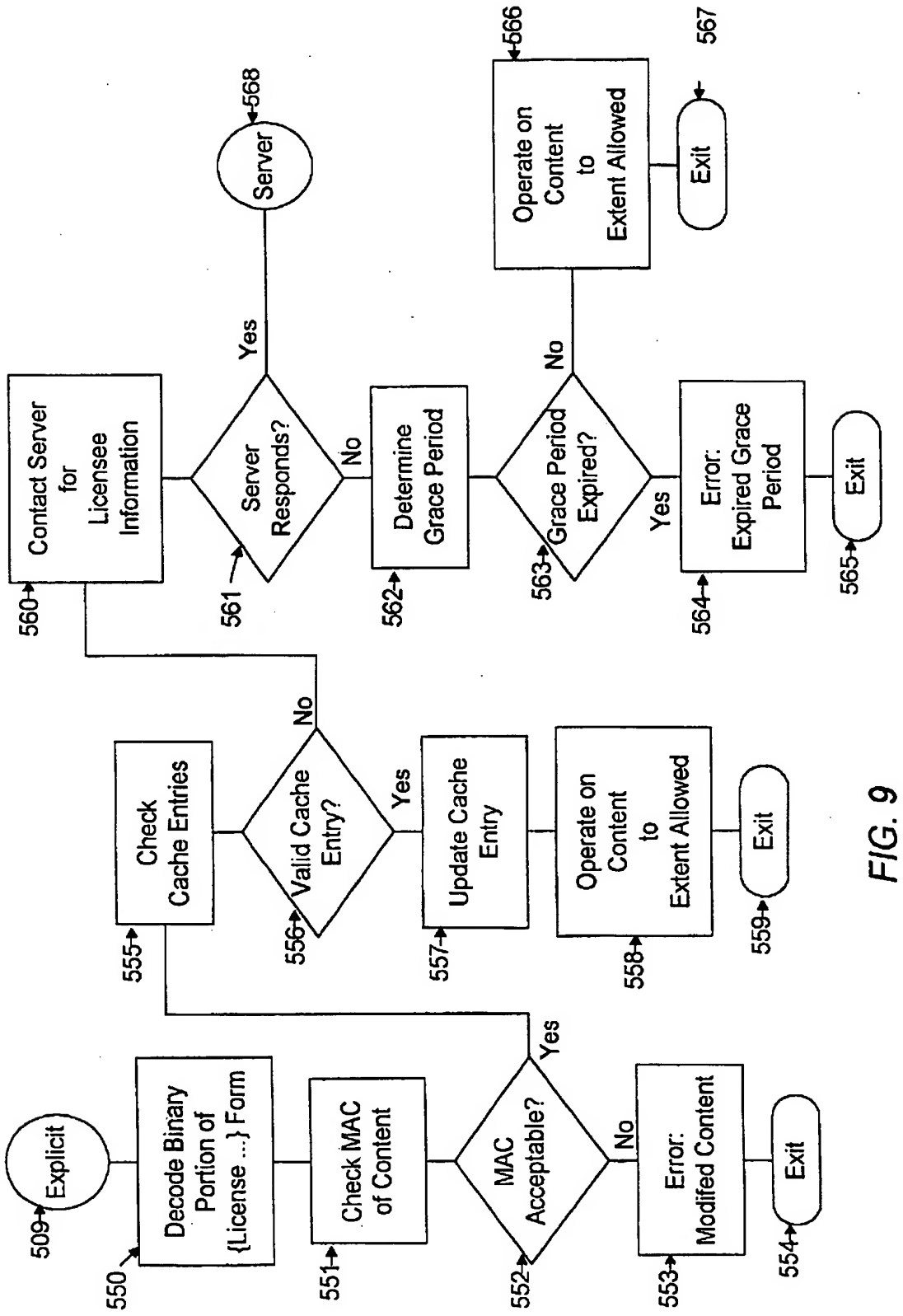

Explicit
509
Decode Binary Portion of {License ...} Form
550
Check MAC of Content
551
MAC Acceptable?
552
Yes
No
Error: Modifed Content
553
Exit
554
Check Cache Entries
555
Valid Cache Entry?
556
No
Yes
Update Cache Entry
557
Operate on Content to Extent Allowed
558
Exit
559
Contact Server for Licensee Information
560
Server Responds?
561
Yes
No
Server
568
Determine Grace Period
562
Grace Period Expired?
563
No
Yes
Error: Expired Grace Period
564
Exit
565
Operate on Content to Extent Allowed
566
Exit
567

FIG. 9

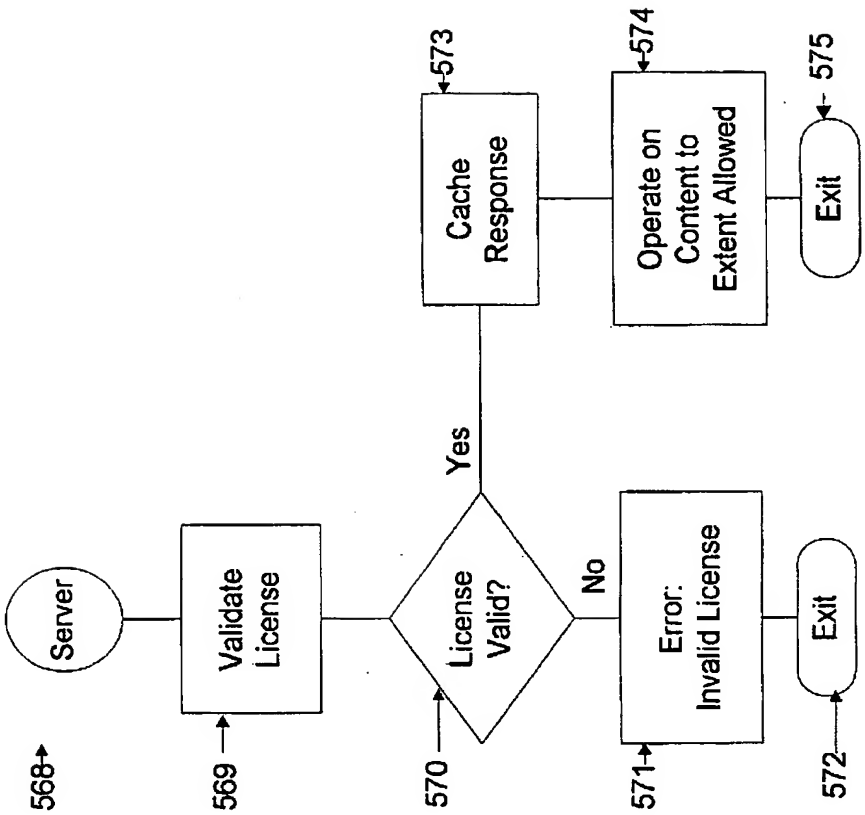

Server
Validate License
License Valid?
Yes
No
Cache Response
Operate on Content to Extent Allowed
Exit
Error: Invalid License
Exit
568
569
570
571
572
573
574
575

FIG. 10

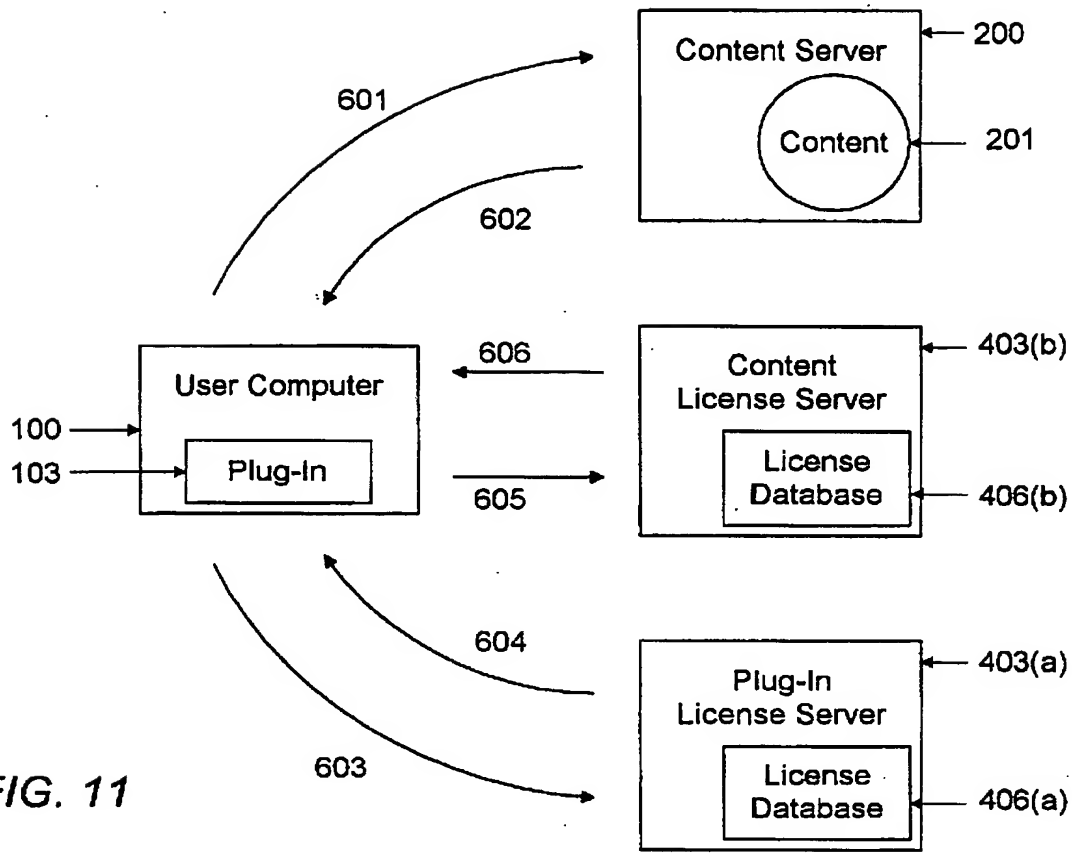

Content Server
200
Content
201
601
602
User Computer
606
Content
License Server
403(b)
100
103
Plug-In
605
License
Database
406(b)
604
Plug-In
License Server
403(a)
603
License
Database
406(a)

FIG. 11

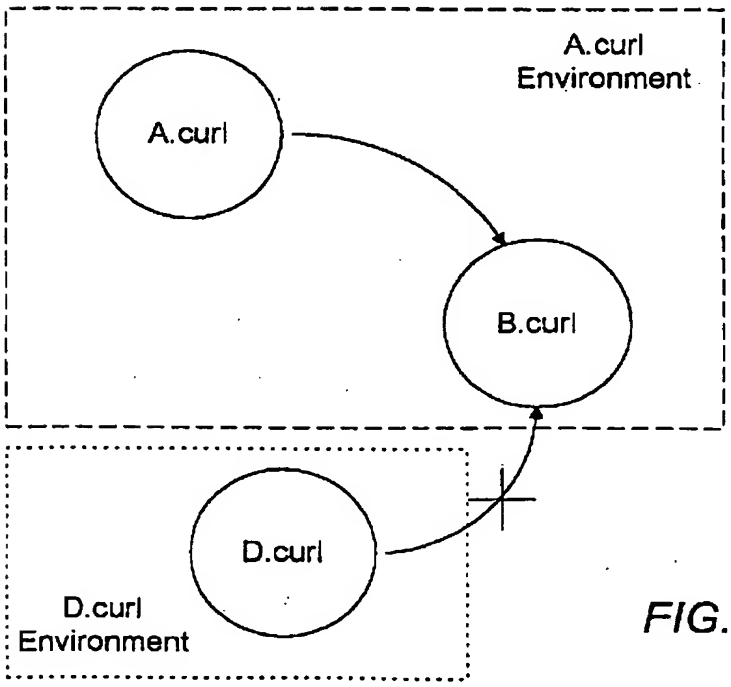

A.curl
Environment
A.curl
B.curl
D.curl
D.curl
Environment

FIG. 12

INTERNATIONAL SEARCH REPORT

Inter inal Application No
PCT/US 00/06242

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 98 25373 A (INTELLECTUAL PROTOCOLS L L C) 11 June 1998 (1998-06-11)	1-3,5-7, 12,13, 15-21, 23-25, 30,31, 33-38, 40,41, 45,46, 48-50
A	abstract page 17, line 14 -page 35, line 20	4,8-11, 22, 26-29, 39,42-44

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 June 2000	06/07/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Jacobs, P

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter inal Application No
PCT/US 00/06242

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	WO 98 45768 A (NORTHERN TELECOM LTD) 15 October 1998 (1998-10-15)	1-3,5, 14-21, 23, 32-38, 40,47-50
A	page 6, line 30 -page 12, line 7 page 24, line 19 -page 25, line 13 figures 1-8	4,6-8, 12,13, 22, 24-26, 30,31, 39,41, 42,45,46

X	EP 0 686 906 A (SUN MICROSYSTEMS INC) 13 December 1995 (1995-12-13)	1,2,4,5, 13-16, 19,20, 22,23, 31-34, 37,38, 40,46-49
A	column 1, line 39 -column 3, line 57 column 9, line 45 -column 13, line 52 figures 4-6	3,17,18, 21,35, 36,39,50

X	US 5 287 408 A (SAMSON PETER R) 15 February 1994 (1994-02-15)	1-5,15, 16, 19-23, 33,34, 37-40, 48,49
A	abstract column 3, line 13 -column 9, line 33 figure 4	12,13, 30,31, 45,46

A	US 5 790 664 A (COLEY CHRISTOPHER D ET AL) 4 August 1998 (1998-08-04)	2,3, 6-13, 15-18, 21,22, 24-31, 33-36, 39, 41-46, 48-50
	abstract column 7, line 43 -column 13, line 28 figures 5-7	

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter: nal Application No
PCT/US 00/06242

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9825373 A	11-06-1998	NONE	
WO 9845768 A	15-10-1998	AU 6492198 A	30-10-1998
		EP 0974084 A	26-01-2000
EP 0686906 A	13-12-1995	US 5724425 A	03-03-1998
		JP 8166879 A	25-06-1996
US 5287408 A	15-02-1994	NONE	
US 5790664 A	04-08-1998	AU 2054597 A	10-09-1997
		WO 9730575 A	28-08-1997